Classificação timbrística à partir da classificação supervisionada

João Teixeira Araújo[1] **, Rômulo Vieira**[1]

[1]Universidade Federal de São João del-Rei

teixeira.araujo@gmail.com, romulo_vieira96@yahoo.com.br

Abstract. *O Music Information Retrieval (MIR) é um campo de estudo concentrado na extração e inferência de características significativas em um determinado som. Esta área possui bastante importância para a implementação de sistemas de busca, recomendação e recuperação de dados, como por exemplo: a busca de uma determinada música em um grande dataset; recomendação musical em sistemas como Spotify e Youtube, e até mesmo a detecção de comandos fornecidos através da voz, como a Cortana, do Windows. Desta forma, o MIR se torna uma área de interesse, apresentando características interdisciplinares que envolve sub-áreas da computação, como a computação musical e aprendizado de máquina, e também áreas como a matemática e música. Desta forma, este trabalho tem como objetivo o desenvolvimento de um sistema de classificação de timbres para distinguir automaticamente o som produzido por uma guitarra e por um baixo. Vale ressaltar que mais relevante que a aplicação em si, são apresentados os estudos das propriedades sonoras, das técnicas usadas para extraí-las de maneira eficiente e de como classificá-las a partir da junção de determinados descritores de áudio e algoritmos de aprendizagem de máquina.*

1 Introdução

O constante avanço tecnológico tem possibilitado diversas mudanças no que diz respeito a área da computação musical. Até então, diversas composições têm sido criadas e gravadas a partir do meio digital, o que acaba criando uma certa necessidade de classificação dos dados de acordo com suas principais características, visto que a quantidade de arquivos pode atingir a casa de milhões e nem sempre o usuário necessita realizar uma pesquisa informando apenas o autor, álbum, ano da canção ou gravadora. Uma das maneiras de se pesquisar uma determinada música seria a busca por metadados, as quais foram implementadas pela empresa Pandora[1], a partir da definição de determinadas *tags* atribuídas por especialistas, e também pela empresa Last.fm[2], onde os próprios usuários definem essas *tags* e as características de uma música.

Roberto Piassi e Marcelo Queiroz [1] definem alguns problemas existentes ao lidar com a busca por metadados, dentre eles: o alto custo para contratar especialistas para analisarem músicas específicas; a lentidão desse processo, devido ao fato de ser manual; as diferentes classificações empregadas por grupos de usuários ou comunidades distintas; a classificação prioritária de artistas mais famosos, que ofusca artistas mais desconhecidos; a incapacidade de capturar completamente o conteúdo musical e cultural das faixas através da classificação por *tags*, devido ao fato de apresentar limitações, e por fim, existir uma dificuldade do público leigo de associar o verdadeiro significado de cada *tag*. Desta forma, a automatização do processo de classificação musical se torna uma área de interessante por fornecer uma base para a organização, recomendação, navegação e busca de sons em bancos de dados enormes.

Levando em consideração o contexto apresentado, este trabalho irá abordar o timbre a partir do ponto de vista científico, realizando uma busca por características objetivas em determinados sinais de áudio que remetem a sons de instrumentos musicais tradicionais, afim de encontrar uma maneira eficiente e sucinta de alterar a sua forma de representação, e consequentemente classificá-la. Isso quer dizer que, a partir de um sinal de áudio como entrada, serão aplicados descritores de áudio para extração de características sonoras, que serão utilizadas em algoritmos de aprendizado de máquina para classificar, de forma automática, os timbres. Vale ressaltar que estas distribuições podem servir para algumas aplicações, como: a busca de sons sob um um timbre específico; alteração e extração de um instrumento em uma música ou até mesmo na procura por uma canção através do reconhecimento de voz, onde o usuário cantarola um determinado trecho de guitarra, bateria ou baixo a fim de achar qual é a música que se encaixa com a entrada fornecida.

2 Análise Sonora

Nesta seção abordaremos o áudio em relação às suas definições e características. Além disso, será explicado o funcionamento de determinados descritores de áudio, que são basicamente as técnicas empregadas para extrair informações sonoras. Por fim, é apresentado o processo de classificação, focado em métodos de aprendizado de máquina para criação de sistemas automatizados, onde o computador aprende a distinguir diferentes tipos de sons para diferentes propósitos.

2.1 Propriedades Sonoras

Antes de tudo, é importante entender alguns conceitos no que diz respeito ao áudio e seu tratamento por meio computacional, a começar pelo som. O som é a propagação de uma compressão ou onda mecânica, transmitida de forma circuncêntrica em meios materiais. Os sons naturais são, na sua maior parte, combinações de sinais representados por uma senoide pura, com velocidade de oscilação ou frequência medida em hertz (Hz) e uma amplitude ou energia medida em decibéis. Os sons audíveis pelo ouvido hu-

[1]http://pandora.com/

[2]http://last.fm/

mano normalmente possuem uma frequência entre 20 Hz e 20000 Hz [2].

Ele ainda pode ser classificado como um som musical, ruído ou palavra falada. O som musical corresponde a pressões acústicas periódicas e frequências múltiplas. Já o ruído, não possui periodicidade bem definida, e sim, um grande número de frequências diversas. A palavra falada, por fim, consiste na mistura de sons ordenados que apresentam periodicidade com diversas frequências vizinhas [3].

Para que o computador interprete o som, é necessário que ocorra o processamento do áudio em questão. Esse é um subcampo do processamento de sinais que se baseia na manipulação eletrônica de sinais de áudio, levando em consideração possíveis representações das ondas sonoras no interior de uma máquina [4]. Dentre os seus fundamentos, está a intensidade do som no ar, que pode ser alterada através de uma modulação ou compressão, reduzindo a faixa dinâmica de ganho do material. Destacam-se também a densidade, responsável pela amplitude de um sinal de áudio, a equalização, o limitador de picos e o ceifamento e compressão multi bandas [5, 6].

Além disso, existem os descritores de áudio, que são ferramentas analíticas que representam características do sinal musical em curvas unidimensionais. Assim, reduzem a complexidade da informação ao focar em aspectos específicos. Os descritores são ainda ferramentas úteis para criar uma taxonomia de particularidades do conteúdo dos espectros do sinal musical [7]. Em suma, eles apresentam caráter reducionista. Esse atributo pode ser correlacionado com propriedades subjetivas da percepção, como “brilho”, “opacidade” ou ainda “maciez” do som [8]. Neste trabalho serão utilizados 3 descritores: Spectral Rolloff, Spectral Centroid e Spectral Flatness, que serão explicados posteriormente.

Quanto a música, ela é uma forma de arte que surge a partir da combinação de vários sons e ritmos, sendo composta por diferentes aspectos, onde cada um deles possui um papel crucial no produto final. Um desses aspectos de maior importância é o timbre, também conhecido como “a cor do tom”. Ao atribuir diferenças nas qualidades sonoras de um instrumento, ele garante a personalidade do som. No sentido técnico, é o responsável por diferenciar uma mesma nota executada em dois instrumentos diferentes, ou até mesmo distinguir um som grave de um agudo [9, 10]. A figura 1 demonstra uma nota sendo executada por diferentes fontes sonoras, sendo elas: diapasão, ferramenta utilizada para afinação de instrumentos, flauta doce, voz humana e violino [11].

Segundo a Sony Europe, esse atributo é resultado da combinação de dois componentes: as vibrações do próprio instrumento e as frequências produzidas por essas vibrações.Importante dizer que cada fonte sonora possui características distintas, ou seja, as formas que elas produzem os sons são diferentes, podendo ser através de dedilhados, percussão, sopro ou por inferência eletrônica [9, 11].

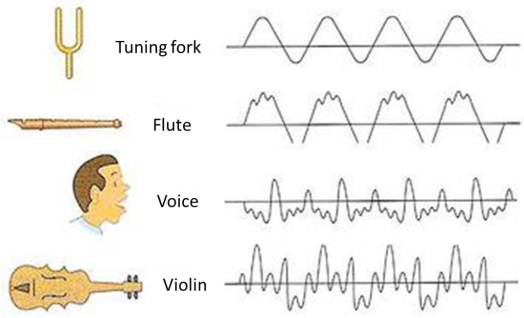

Figura 1: Exemplos timbrísticos.

2.2 Extração de Características Sonoras

Atualmente, existem diversas pesquisas em relação ao desenvolvimento de ferramentas tecnológicas para a aplicação de descritores de áudio, seja na forma de bibliotecas para programação, ou na forma de *softwares* gráficos e extensões para extração e visualização das características de um áudio. Dentre as ferramentas existentes, destacam-se: bibliotecas Aubio [12] e LibXtract [13]; frameworks como MARSYAS [14], jAudio [15], CLAM Music Annotator [16], jMIR [17] e Sonic Visualiser [18]; e finalmente, a toolbox para MatLab, MIRtoolBox [19].

O primeiro dos descritores a serem utilizados neste trabalho é o Spectral Rolloff, que extrai o ponto de rolagem, ou seja, os segmentos da onda sonora que estão abaixo de determinada percentagem. Um exemplo deste descritor pode ser observado na figura 2 [20, 21].

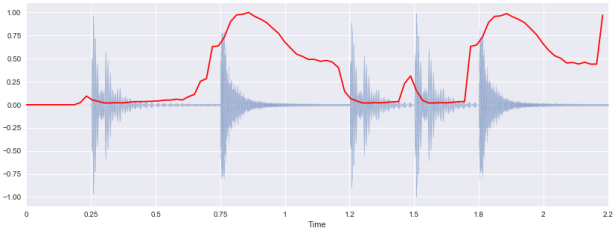

Figura 2: Curva do espectral do descritor Rolloff.

O segundo descritor utilizado foi o Spectral Centroid, responsável por indicar onde está o centro de massa do espectro, isso é, a tendência central da forma de onda, assim como mostra a figura 3. Ele tem uma conexão com a impressão de “brilho” do som, e suas aplicações mais comuns são voltadas justamente para a classificação de timbres.

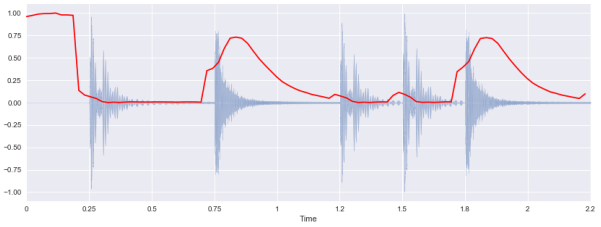

Figura 3: Curva espectral do descritor Centroid.

O terceiro descritor foi o Spectral Flatness,

também conhecido como coeficiente de tonalidade ou Entropia de Wiener. É uma medida usada para quantificar a aparência de um ruído, em oposição ao tom, assim como mostra a Figura 4 [22, 23, 24].

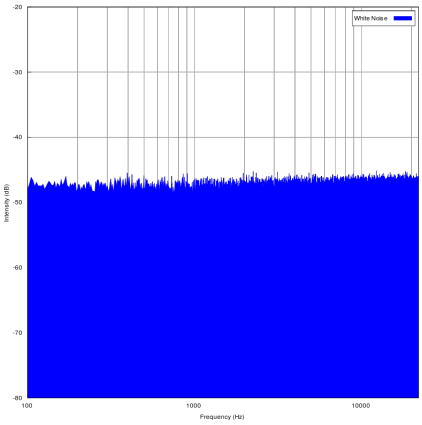

Figura 4: Curva espectral do descritor Flatness.

Por fim, o último e mais importante descritor, o Mel Frequency Cepstral Coefficient (MFCC). Ele é um derivado da representação de um espectro de áudio não-linear, muito utilizado em sistemas de reconhecimento de fala, classificação de gêneros musicais e medidas de similaridade de áudio [25]. Merece destaque o fato do MFCC não apresentar robustez diante de ruído, portanto, seus valores devem ser normalizados [26]. A Figura 5 exibe uma amostra de áudio após a aplicação do MFCC.

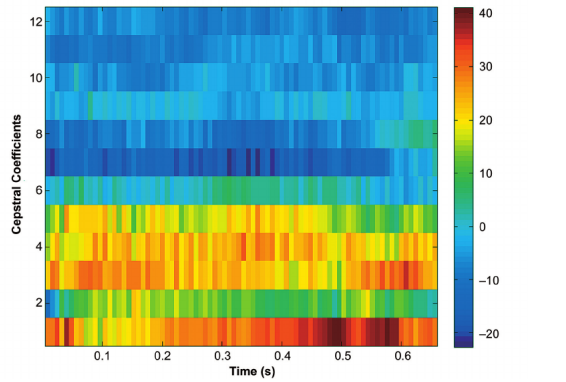

Figura 5: Espectrograma (MFCC).

2.3 Classificação Sonora

O processo de classificação sonora é obtido através da aplicação de técnicas de aprendizado de máquina. Ou seja, a partir de uma base de dados pré-processada, o computador segue um conjunto de passos que irão ensiná-lo a diferenciar dois áudios distintos. Estas técnicas podem ser úteis para solucionar problemas, uma vez que a máquina é mais eficiente do que um humano para classificar grandes quantidades de dados ou até mesmo para realizar predições sobre eles.

Neste trabalho, o Naive Bayes foi escolhido para classificação automática dos dados. Este algoritmo é um classificador probabilístico, baseado no teorema do matemático Thomas Bayes. O Teorema consiste em multiplicar uma probabilidade *a priori*, as quais são coletadas de maneira prévia, por uma probabilidade *a posteriori*, obtida após a ocorrência do evento. Esta técnica se baseia na independência condicional de valores, onde a presença de determinada *feature* não tem correlação com nenhuma outra [27]. A equação referente ao Teorema de Bayes pode ser visualizada abaixo.

$$P(A \mid B) = \frac{P(B|A)P(A)}{P(B)}$$

Em relação à equação, P(A) e P(B) são as probabilidades *a priori* de seus respectivos eventos. P(A|B) é a probabilidade *a posteriori* de A, dado que um evento B ocorreu, e P(B|A) é a ocorrência de B após A.

O funcionamento do algoritmo é intimamente ligado a equação, baseando-se na multiplicação de eventos *a priori* com eventos *a posteriori*. Ou seja, ele computa a probabilidade de uma amostra pertencer a uma determinada classe a partir das chances *a priori* referentes a ela estar contida na classe analisada. Também são calculadas as probabilidades condicionais de cada exemplo ocorrer na mesma classe. Assim, seu objetivo é encontrar a melhor associação para um arquivo de entrada [28].

Em seguida, os dados são normalizados e classificados de acordo com as classes que compõem o problema. Sua principal vantagem é necessidade de um pequeno número de dados de treinamento para estimação dos valores. É considerado um dos métodos mais efetivos hoje em dia, atuando em problemas do mundo real, mesmo que sua construção seja relativamente simples [29, 30].

É importante ressaltar que não existe apenas uma forma de implementação para o Naive Bayes. Na literatura, existem diversas outras aplicações, como filtro de *spam* para e-mails, classificação de notícias de acordo com seu conteúdo, diagnósticos de doenças, entre outros. A figura 6 exibe uma classificação de dados genéricos utilizando esse algoritmo.

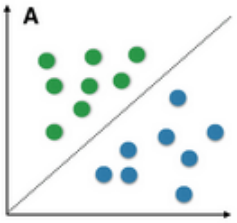

Figura 6: Classificação de dados após aplicado um algoritmo de Naive Bayes.

Para o problema da classificação automática de timbres, seria como se os círculos verdes representassem os samples de uma guitarra e os azuis, os de baixo. Neste caso, o Naive Bayes teria classificado os áudios com uma acurácia de 100%, o que nem sempre é comum na aplicação de algoritmos de aprendizagem de máquina.

3 Ferramentas Utilizadas

Esta seção aborda todas as ferramentas utilizadas para o desenvolvimento do trabalho. Elas vão desde as bibliotecas inseridas até as *frameworks* que forneceram apoio para implementação do sistema de classificação, assim como a base de dados usada no trabalho.

3.1 Python 3.7.4

Python é uma linguagem de programação de alto nível, interpretada, imperativa, orientada a objetos, funcional e de tipagem econômica e forte. Criada por Guido van Rossum, em 1991, possui um modelo de desenvolvimento comunitário, aberto e sem fins lucrativos. Embora apresente aspectos formais, ela não foi especificada [31, 32]. A linguagem é procedural e dinâmica, sendo de fácil leitura de código, além de exigir poucas linhas para executar alguma função, comparada as demais linguagens. Em relação ao tratamento de áudio, o Python é umas das linguagens mais utilizadas nesse campo.

3.2 PyCharm

Lançado em julho de 2010, pela empresa tcheca JetBrains, o PyCharm é uma *Integrated Development Environment* (IDE) usada para programação, em especial na linguagem Python. A ferramenta provém análise de código, depurador gráfico, unidade de teste integrada e ainda permite a conexão com outros sistemas de controle. É uma aplicação multiplataforma, disponível para Windows, macOS e Linux [33, 34], permitindo também a extensão de funcionalidades através de *plugins*, contando hoje com mais de 1000 disponíveis. Além disso, tem destacamento de erros e sintaxe, correções rápidas, navegação versátil entre o código e suporte para *frameworks web*. Foi escolhida para o projeto por ser uma das melhores alternativas na programação em Python.

3.3 Jupyter-Notebook

Criado em 2014, por Fernando Pérez, é um subproduto do IPython e foi desenvolvido para para propiciar o desenvolvimento de *softwares* em código aberto em diversas linguagens de programação. Já o Notebook Jupyter é um ambiente computacional *web* para criação de documentos nessa plataforma [35]. Para o projeto em questão, foi utilizado para criação do algoritmo de Naive Bayes e para análise dos espectrogramas gerados.

3.4 Bibliotecas

A biblioteca Librosa é um pacote em Python, desenvolvido para áudio e processamento digital na música. O produto tem o objetivo de facilitar o trabalho em MIR. Apresenta familiaridade com MATLAB, padronização de comandos e característica modular, permitindo que o programador aumente as capacidades de trabalho da biblioteca [36]. Algumas funções merecem destaque, como a librosa.beat, que permite o uso de funções para estimar o tempo e detectar batidas, librosa.core, que inclui funções para carregar arquivos de áudio, representações de espectrograma ou outra informação musical e librosa.display, responsável por visualizações gráficas integradas ao MATLAB (Figura 7 [37]).

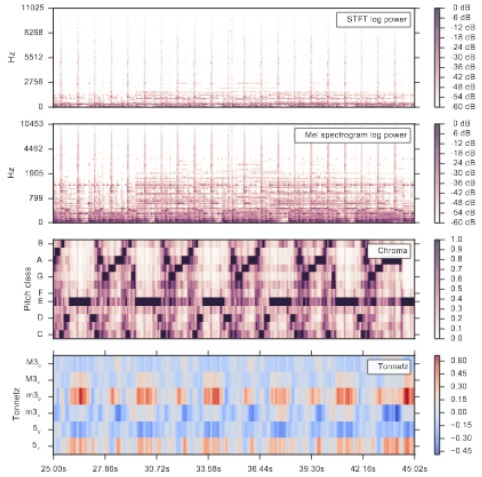

Figura 7: Espectrogramas tratados na biblioteca.

Já a AudioLib, criada por Jacob Tsekrekos, é menos potente que o LibRosa, entretanto fornece um sistema de alta qualidade para executar áudio em aplicações Python, sendo sua principal característica a capacidade de gerar formas de onda, o que é essencial para a diferenciação de timbres [38].

Quanto a biblioteca para classificação dos dados, a escolhida foi a scikit-learn, também conhecida como Sklearn. Lançada em 2007, desde então tem sido utilizada para identificar a categoria e o valor contínuo de um objeto, automatizar o agrupamento de valores familiares, extrair e normalizar características e muitas outras tarefas. Ainda tem a vantagem de ser projetada para utilizar tecnologias consolidadas no campo da Inteligência Artificial, como NumPy, Matplot e Pandas [39, 40].

3.5 NSynth Dataset

Considerado referência no estudo da síntese de áudio, o Nsynth é um banco de dados de grande magnitude, com cerca de 20 GB e 305.979 notas musicais, cada uma com um timbre, altura e envelope exclusivo. Essas notas possuem três informações adicionais: fonte, família e qualidade. A fonte indica o método de produção do som (nesse caso, acústico, eletrônico ou sintetizado), enquanto a família é responsável pela classificação dos instrumentos de acordo com suas características e a qualidade separa as amostras a partir da agradabilidade sonora [41].

O banco conta ainda com amostras de mais de mil instrumentos comerciais e sons em diferentes velocidades, o que garante a eles variadas propriedades [41]. Disponível nos formatos TFRecord e JSON (Figura 8), foi utilizado em sua versão mais atual, que data de 2017.

Além da classificação timbrística, de acordo com a feature referente à fonte sonora, podemos realizar a extração de características sonoras para realizar a distinção entre instrumentos tocados normalmente por alguém e ins-

```
"bass_synthetic_033-022-050": {
    "note": 201034,
    "sample_rate": 16000,
    "instrument_family": 0,
    "qualities": [
        0,
        1,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0
    ],
    "instrument_source_str": "synthetic",
    "note_str": "bass_synthetic_033-022-050",
    "instrument_family_str": "bass",
    "instrument_str": "bass_synthetic_033",
    "pitch": 22,
    "instrument": 417,
    "velocity": 50,
    "instrument_source": 2,
    "qualities_str": [
        "dark"
    ]
}
```


Figura 8: NSynth - Json fornecido (NSynth).

trumentos produzidos sinteticamente, através do computador. Outra feature interessante é a referente à qualidade do som Figura 9).

Index	ID	Description
0	bright	A large amount of high frequency content and strong upper harmonics.
1	dark	A distinct lack of high frequency content, giving a muted and bassy sound. Also sometimes described as 'Warm'.
2	distortion	Waveshaping that produces a distinctive crunchy sound and presence of many harmonics. Sometimes paired with non-harmonic noise.
3	fast_decay	Amplitude envelope of all harmonics decays substantially before the 'note-off' point at 3 seconds.
4	long_release	Amplitude envelope decays slowly after the 'note-off' point, sometimes still present at the end of the sample 4 seconds.
5	multiphonic	Presence of overtone frequencies related to more than one fundamental frequency.
6	nonlinear_env	Modulation of the sound with a distinct envelope behavior different than the monotonic decrease of the note. Can also include filter envelopes as well as dynamic envelopes.
7	percussive	A loud non-harmonic sound at note onset.
8	reverb	Room acoustics that were not able to be removed from the original sample.
9	tempo-synced	Rhythmic modulation of the sound to a fixed tempo.

Figura 9: Feature sobre qualidade sonora.

Este tipo de feature nos fornece informações bastante interessantes para criar outros tipos de classificações, como por exemplo a distinção entre o que seria um som tocado sob o efeito de distorção e o que seria um som mais "limpo". Vale ressaltar que o valor de "percussive" é extremamente importante caso o objetivo do trabalho fosse o de classificar uma bateria de outro instrumento qualquer, visto que a presença de notas não-harmônicas de alta amplitude no início de uma amostra / sample pode distinguir com facilidade um instrumento percussivo de um outro instrumento qualquer.

3.6 Árvore de Decisão

A Árvore de Decisão é uma estrutura hierárquica caracterizada pelo aprendizado supervisionado. Sendo assim, os dados de entrada têm suas características definidas desde o começo e usa esse conhecimento para atuar sobre os novos dados que são recebidos [42].

As Árvores, geralmente utilizadas na área da computação, são formadas por nós, galhos e folhas. Os nós representam os atributos, os galhos dizem respeito às regras de classificação, e as folhas referem-se aos resultados. Cada escolha por um segmento diferente irá culminar em resultados distintos [42].

O objetivo do algoritmo é dividir progressivamente o *dataset*, particionando o espaço em sub-regiões com base em um recurso descritivo. Essa redução do espaço acontece até que eles sejam tão pequenos que possam ser definidos por um único rótulo. A partir disso, os novos dados de entrada também são classificados por um rótulo. Para o caso em questão, as amostras foram definidas como baixo ou guitarra [43].

Para ilustrar o funcionamento, temos a própria classificação de timbres. Suponha que uma nova amostra de áudio foi inserida no sistema e portanto, deve ser designada como baixo ou guitarra. A primeira pergunta que o algoritmo deve responder é qual a frequência dessa amostra, que corresponde ao nó-raiz da árvore de decisão. A seguir, outras perguntas podem ser feitas, utilizando como base a resposta anterior. Esta sequência de questionamentos e respostas são organizadas de modo a gerar uma árvore, que irá indicar o caminho a ser seguido para se alcançar uma resposta. Para isso, basta partir do nó-raiz da árvore e percorre-la até um nó folha, responsável por indicar a classe daquele áudio [43].

Algumas informações são de vital importância para a construção do método. A primeira delas, é o ganho de informação. O ganho representa o que foi aprendido sobre os rótulos quando uma região é dividida em duas sub-regiões, de acordo com o ponto de corte, como na fórmula abaixo [43, 44]:

$$Gain(R, R_e, R_d) = H(R) - \frac{|R_e| * H(R_e) + |R_d| * H(R_d)}{|R|}$$

Onde H indica a impureza da região R, Re é a sub-região da esquerda e Rd a sub-região da direita.

O segundo fator de destaque é a entropia, responsável por dizer o grau de impureza e instabilidade do classificador. Para isso, ela utiliza-se do nó pai e do nó filho. Aquele atributo que gerar maior diferença, será o escolhido. Essa condição pode ser observada a seguir [44].

$$entropia(R) = -\sum p(c|R) log(p(c|R))$$

P(c|R) indica a probabilidade de um ponto da região R pertencer a classe c. Essa probabilidade é estimada pela razão entre a quantidade desses pontos.

O último atributo é o fator Gini, baseado na teoria de dispersão estatística criada pelo matemático Corrado Gini, sendo um índice que retrata esta dispersão. Desta forma, o maior índice é escolhido para integrar o sistema. A equação que ilustra essa teoria pode ser visualizada abaixo[44]:

$$gini(R) = \sum p(c|R)(1 - p(c|R))$$

Para esta condição as variáveis c e R assumem os mesmos valores para a entropia.

A Figura 10 retrata uma Árvore de Decisão. O nó cinza representa a pergunta inicial, ou seja, o nó-raiz. Os galhos são os círculos azuis e os triângulos alaranjados são as folhas [45].

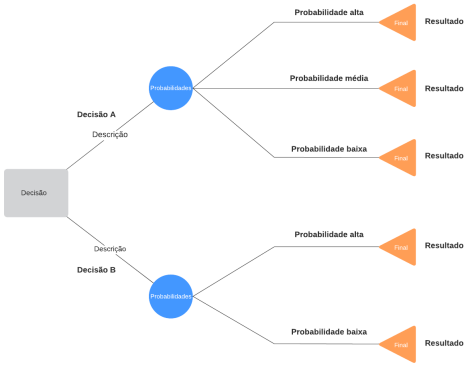

Figura 10: Modelo padrão de uma Árvore de Decisão.

3.7 Support Vector Classifier

Antes de ser abordada as características e aplicações do Support Vector Classifier (SVC), é necessário entender qual técnica deu origem a ele. Essa técnica é chamada de Support Vector Machine (SVM), ou seja, um conceito da computação que analisa dados e reconhece padrões comuns em sistemas de classificação e regressão. O modo de operação consiste em analisar um dado de entrada e alocá-lo em uma das duas classes disponíveis [46].

Com o intuito de otimizar o SVM, surge o Support Vector Classifier. Essa otimização se dá a partir de um ajuste dos dados fornecidos, aplicando a eles penalidades, funções de perda, pontuações de métricas e limites de decisão para verificar qual terá um melhor ajuste ao problema analisado. Em seguida, os dados com melhor desempenho são transportados para um hiperplano, onde serão alimentados com alguns recursos de melhoria para decidir as classes dos objetos de entrada [47, 48].

Uma das vantagens a ser destacada é o fato de ser um esquema um-contra-um, o que significa que cada classificador é comparado diretamente com outro classificador. Também podem ser citadas a *hinge loss*, uma função de classificação a partir da margem máxima e a melhor escalabilidade das escolhas de penalidade em amostras grandes [49, 50].

Apesar de apresentar algumas desvantagens, como não punir interceptação e convergir em um tempo mais lento comparado a outras técnicas, o SVC foi utilizado nesse trabalho para classificar as amostras de baixo e guitarra.

3.8 Multiclassificadores

Os Multiclassificadores consistem na combinação e aplicação de um conjunto de algoritmos de classificação nas partes do espaço de entrada onde eles apresentam melhor desempenho, com o objetivo de otimizar os valores de saída. Esse tipo de sistema é caracterizado por permitir que dados considerados fracos consigam oferecer bons resultados e por não exigir um ajuste tão preciso da base [51].

As técnicas podem ser divididas em dois tipos de grupos: sequencial e paralelo. Como o nome indica, o sequencial é onde o aprendizado da base de dados acontece de forma linear. Sua motivação básica é explorar as dependências entre os dados e obter melhora a partir dos que foram rotulados de forma errônea. Trabalhando de forma oposta ao sequencial, o paralelo explora a independência entre os valores [52].

Na classificação de timbres de baixos e guitarra, a técnica escolhida foi a aplicação do Bagging um pouco mais simplificado. O procedimento consiste em escolher uma amostragem aleatória do conjunto de dados e gerar novos subconjuntos. Em cada uma dessas divisões, é aplicado um classificado diferente, para que o algoritmo obtenha diversidade nos modelos de classificação. Uma vez que o procedimento é encerrado. ele inicia um processo de votação simples para determinar qual das classes deve ser escolhida. A classe selecionada é que obteve maior quantidade de respostas corretas para diferentes classificadores [53]. Vale ressaltar que neste trabalho a base de treino e teste foi aplicada igualmente para os 3 classificadores, e não um subconjunto para cada um, como no Bagging.

3.9 Cross Validation

Uma vez terminado o treinamento do modelo, é difícil ter a certeza que ele está funcionando bem para dados desconhecidos. Em outras palavras, não é possível dizer se os resultados são bons diante das variações do processo [54].

É nesse contexto que surge a aplicação da técnica de Cross Validation. Com o objetivo de avaliar a qualidade do classificador, ela divide o conjunto de dados em duas classes: teste, para treinar o modelo, e validação, para informar o quão bom é o algoritmo na generalização dos dados. Esse mecanismo é ideal para evitar que o algoritmo ”decore“ quais resultados são aceitáveis, obrigando-o assim, a aprender a melhor solução para o problema [55].

Para a primeira parte do trabalho, a aplicação de Cross Validation se deu a partir do holdout, uma de suas técnicas para divisão do banco de dados. Ela consiste em separar uma percentagem das amostras para treino, determinada parte para teste, onde a fatia correspondente ao treino deve ser maior. Esse recurso é bastante indicado para bases que contenham muitos valores, para assim evitar variações drásticas no erro [56].

A figura 11 demonstra como a base é divida. Aqui foi determinado que 60% dos dados eram destinados para treino e 40% para teste.

Ainda sobre os métodos de Cross Validation aplicados nesse trabalho, vale citar o K-Fold, empregado na separação de dados para a segunda parte. Além de ser popular, apresenta uma fácil compreensão e gera resultados menos tendenciosos que outros modelos, já que garante que todos os conjuntos de dados apareçam como teste e como treino [55].

Esse recurso consiste em dividir o conjunto de dados em K pastas. É importante dizer que o valor de K não deve ser muito alto nem muito baixo, com intervalo ideal entre 5 e 10 divisões. Feito isso, uma pasta é escolhida para teste, enquanto as restantes são usadas para treino. O processo se repete até que todas as pastas tenham servido como modelo de teste por uma única vez (Figura 12) [55, 57].

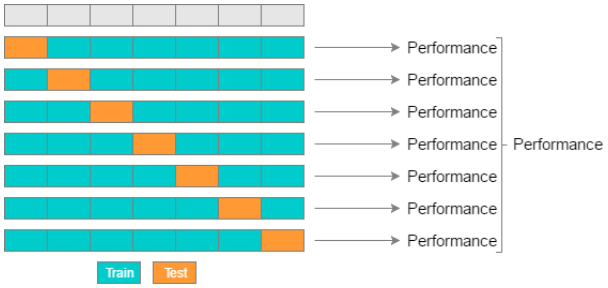

Figura 12: Divisão em subconjuntos do K-Fold.

No trabalho em questão, o número de pastas escolhidas foi 5 e verificou se o algoritmo aprendeu bem como fazer a distinção entre *samples* de baixos e guitarras e garantir que os dados não estavam escassos nem em *overfitting*.

3.10 F-test

Em um problema de classificação a mineração de dados tem um papel tão importante quanto o próprio sistema de inteligência artificial escolhido para essa função. Dentre os benefícios de uma boa seleção de dados, encontra-se: evitar o *overfitting*, melhorar a performance e reduzir o tempo de treinamento [58].

Um dos métodos para fazer a seleção de *features* é o F-test. Ele é um tipo de teste estatístico usado para identificar o melhor modelo que se adapta a uma população. Essa identificação é feita através de um confronto entre os erros dos modelos distintos [58, 59]. A cada teste de modelo é aferido um valor de importância para cada *feature*, de acordo com sua melhora ou piora atribuída ao sistema.

Na classificação de timbres, o F-test é usado após a divisão do *dataset* pelo Cross Validation. Logo após, ele testa as duas melhores *features* e repete a classificação até que encontre as 3, 5, 6 e 8 melhores.

Figura 11: Divisão em subconjuntos do holdout.

4 Pré-processamento

Inicialmente, foram coletados 2500 *samples* distintos, com diferentes frequências, referentes a guitarra e baixo. Através da biblioteca Librosa, extraiu-se os valores numéricos de cada *sample*, visto que um áudio/onda sonora pode ser representado por um vetor, os quais são usados para reconstruir uma onda sonora a partir de seu *sample rate* e da aplicação de equações trigonométricas.

Usando este vetor como base, aplicou-se a Short-Time Fourier Transform (STFT). A STFT é uma transformada matemática integral aplicada em funções que variam com o tempo, chamadas de não-estacionárias. A técnica consiste na divisão do espectrograma em intervalos menores, fazendo com que cada um deles se torne uma constante. Posteriormente, é aplicada uma transformada de Fourier em cada uma dessas repartições. O método foi utilizado para coletar o módulo do resultado do vetor, já que ele possui valores negativos e positivos.

Após aplicar o módulo, a saída da STFT é utilizada nos descritores de áudio, já explicados na subseção 2.2, que irão oferecer um outro vetor numérico com determinadas características do áudio. A partir desse vetor, é aplicada a média e o desvio-padrão entre cada um, fornecendo então duas *features* por descritor.

Desta forma, os dados de cada sample estão tratados e prontos para serem utilizados para classificação. A Figura 13 demonstra como ficaram os dados antes da aplicação do algoritmo Naive Bayes. Em relação aos dados pré-processados, são 8 *features* para cada *sample*. Além disso, é retornada a classe referente ao *sample*, podendo ser 0 para baixo e 1 para guitarra.

Sample 1	média (centroid)	desvio (centroid)	média (flatness)	...	Classe1
...	...	...	...	...	...
Sample N	...	...	...	...	Classe N

Figura 13: Dados pré-processados

Um apontamento que merece destaque é que não foram encontrados problemas referentes a dados inconsistentes ou faltantes.

5 Resultados

Esta seção irá abordar todos os resultados desta pesquisa, tanto os resultados obtidos inicialmente, à partir da comparação entre o Naive Bayes implementado e o fornecido pela Sklearn, quanto as melhorias e técnicas utilizadas para melhorar a acurácia da classificação timbrística.

5.1 Resultados Preliminares

Inicialmente, foram aplicados os algoritmos sobre 6 *features* coletadas pelos descritores Centroid, Rolloff e Flatness. Em relação ao classificador fornecido pela Sklearn, foi obtido uma acurácia de 76,7% (Figura 14).

Analisando os dados mais afundo, sobre os 1000 *samples* de teste, nota-se que o algoritmo foi melhor em classificar amostras de guitarras, errando em apenas 187

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()

# Train classifier
classifier.fit(X_train, y_train)

# Run prediction
y_pred = classifier.predict(X_test)

# Evaluate
acc = accuracy_score(y_test, y_pred)
print(acc)
```

```
0.767
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# conf[i,j]: true is i, predicted is j
conf = confusion_matrix(y_test, y_pred)
print(conf)
recall = classification_report(y_test, y_pred)
print(recall)
```

```
[[721 279]
 [187 813]]
              precision    recall  f1-score   support

           0       0.79      0.72      0.76      1000
           1       0.74      0.81      0.78      1000
```

Figura 14: Resultado Naive Bayes (Sklearn) - 3 descritores

samples, sendo que para o baixo foram 279 amostras classificadas erroneamente. Essa diferença acabou influenciando no f1-score, que é uma métrica que combina a precisão e o *recall* obtido para analisar a qualidade geral do modelo. Desta forma, o f1-score foi de 0,78 para guitarra e 0,76 para baixo.

Já o Naive Bayes implementado, obteve uma acurácia um pouco pior, acertando 76,2% da base de teste, errando 273 *samples* de guitarra e 203 de baixo.

Depois de aplicar os três descritores, o MFCC foi adicionado as *features*, totalizando então 8 características para a aplicação do Naive Bayes. O algoritmo da Sklearn respondeu positivamente a adição do MFCC, aumentando a eficácia de 76,7% para aproximadamente 79,8% (Figura 15).

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()

# Train classifier
classifier.fit(X_train, y_train)

# Run prediction
y_pred = classifier.predict(X_test)

# Evaluate
acc = accuracy_score(y_test, y_pred)
print(acc)
```

```
0.7985
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# conf[i,j]: true is i, predicted is j
conf = confusion_matrix(y_test, y_pred)
print(conf)
recall = classification_report(y_test, y_pred)
print(recall)
```

```
[[727 273]
 [130 870]]
              precision    recall  f1-score   support

           0       0.85      0.73      0.78      1000
           1       0.76      0.87      0.81      1000
```

Figura 15: Resultado Naive Bayes (Sklearn) - 4 descritores

O algoritmo implementado obteve uma taxa de melhoria semelhante, se comparado ao algoritmo da Sklearn. Após adicionar o MFCC, a precisão melhorou de 76,2% para aproximadamente 79,5%, obtendo um resultado bem próximo do algoritmo da Sklearn.

Visando uma análise mais aprofundada sob a influência dos descritores no resultado final dos classificadores apresentados, a matriz de correlação entre as *features* foi criada (Figura 16).

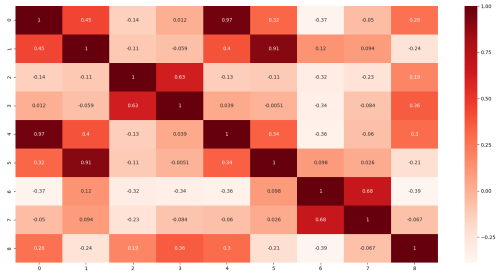

Figura 16: Matriz de correlação entre as features

Porém, por mais que a matriz demonstre todas as correlações entre as *features*, a tarefa de definir quais foram realmente relevantes ainda foi um pouco difícil. Para isso, levando em consideração as classes dos *samples*, foi realizado um ranqueamento (Figura 17).

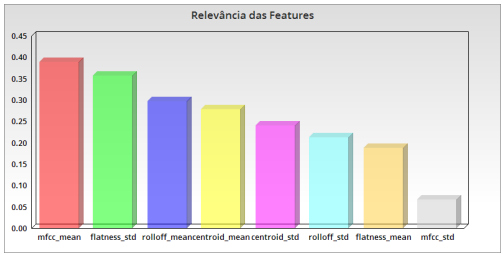

Figura 17: Ranking de features

A partir do ranking gerado, nota-se que a adição do MFCC foi de extrema importância para a melhoria dos resultados, visto que a *feature* baseada na média dele foi a mais relevante entre todas as outras utilizadas. Em vista que nem sempre a média e o desvio-padrão serão relevantes para a classificação de timbres, o MFCC também teve um comportamento interessante neste ponto, já que sua média resultou no atributo mais relevante e o seu desvio-padrão na mais desprezível.

Outro comportamento que pode ser analisado é que em nenhum dos casos a média e o desvio-padrão de um mesmo descritor foram relevantes ao mesmo tempo. Quando a média advinda de um descritor era relevante, seu desvio-padrão resultava em uma *feature* não muito interessante, e vice-versa. Na Figura 17 nota-se que as 4 primeiras características são referentes a cada descritor utilizado, seja pela média ou pelo desvio-padrão. Isso indica que, para cada descritor, apenas um dos dois foi relevante, mas nunca os dois em conjunto.

5.2 Resultados Posteriores

Os primeiros resultados obtidos foram de extrema importância para compreensão sobre como criar um sistema

de classificação de timbres. Na etapa anterior, não foram considerados a eficiência e eficácia das etapas de pré-processamento, particionamento dos dados e também a própria acurácia da classificação realizada.

A fim de melhorar a classificação em geral, inicialmente foi implementado um sistema simples de arquivos, responsável por armazenar os dados pré-processados, ou seja, todos os descritores de áudio foram aplicados uma única vez e então armazenados em um arquivo. Desta forma, a aplicação de testes sob diferentes algoritmos de classificação ficou mais simples e rápida.

Após o sistema de armazenamento dos dados pré-processados, foi realizada uma melhoria sob a forma de particionamento dos dados em treino e teste. Anteriormente a base era simplesmente dividida entre 60% para treino e 40% para teste, o que parece ser bastante trivial e nem sempre eficaz. Desta forma, foi implementado a técnica de Cross-validation, K-Fold, explicada na subseção 3.9, com o parâmetro K valendo 5. Ou seja, para cada teste referente a um classificador, o algoritmo fará 5 "sub-testes" dividindo a base de dados entre 1000 samples para teste e 4000 para treino, realizando ao final, uma média entre os 5 para definir a acurácia geral obtida.

Além disso, ao invés de retirar cada *feature* manualmente para uma possível melhoria dos resultados, foi aplicada uma técnica mais eficaz, chamada F-Test, explicada na subseção 3.10. Desta forma, o F-Test foi aumentando gradativamente a quantidade dos atributos mais relevantes. Ou seja, primeiro foi realizado um teste com 20% das melhores*features*, seguido das 40%, 60%, 80% E 100%. Ou seja, 5 testes contendo as 2, 3, 5, 6 e 8 melhores características coletadas.

Por fim, em relação às melhorias realizadas sob o classificador utilizado, foi aplicada a técnica de ensemble sob um multiclassificador, explicados na subseção 3.8. Neste trabalho, uniu-se os seguintes classificadores: Naive Bayes implementado; Decision Tree fornecido pela Sklearn; e o Suport Vector Machine. Ao coletar os resultados obtidos pelos classificadores, um sistema de voto é então realizado, onde para cada instância, a classe mais votada é a escolhida.

Após implementar os multiclassificadores, os testes foram realizados (Figura 18).

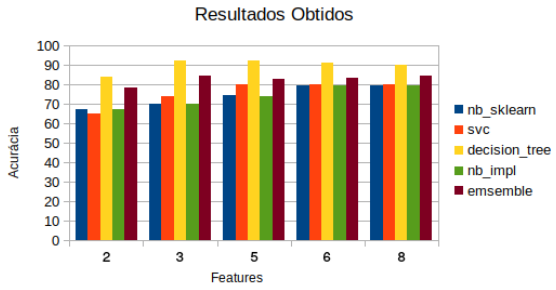

Figura 18: Resultados obtidos

No gráfico acima, temos então os 5 testes realizados, dentre eles os testes com 2, 3, 5, 6 e 8 *features*, realizados sob os classificadores: Naive Bayes, Support Vector Machines e árvores de decisão, ambos fornecidos pela Sklearn; Naive Bayes implementado; e por fim, o multiclassificador também implementado.

Analisando os dados mais afundo, observamos que ambos classificadores foram melhorando gradativamente a acurácia de acordo com o aumento de *features*. Vale ressaltar que a árvore de decisão foi o classificador que obteve melhor resultado, variando entre aproximadamente 85% e 90%.

Como o objetivo deste trabalho é a comparação entre o Naive Bayes fornecido pela Sklearn e o classificador implementado, para uma melhor análise temos então a Figura 19 demonstrando tanto os resultados obtidos anteriormente, referentes ao Naive Bayes implementado e o fornecido pela Sklearn, quanto os resultados do multiclassificador.

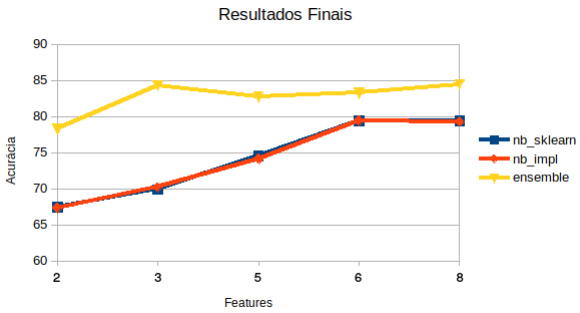

Figura 19: Comparação entre os algoritmos

De acordo com o gráfico, o Naive Bayes implementado e o fornecido pela Sklearn obtiveram resultados e comportamentos bastante semelhantes, se diferindo um pouco apenas quando o número de features foi de 3 e 5. Para 3 features, o algoritmo implementado teve uma acurácia um pouco melhor, se comparado ao fornecido pela Sklearn, já para 5 features, o resultado foi o contrário.

Já o multiclassificador, obteve uma acurácia bastante eficaz se comparada aos demais, variando entre aproximadamente 75% e 80%. Vale ressaltar, que nem sempre o aumento de *features* irá resultar na melhoria da acurácia. Se observarmos os testes realizados pelo multiclassificador com 3, 5 e 6 *features*, a acurácia obtida com 3 foi muito maior do que com 5 e 6, sendo igualada apenas com o uso de 8 *features*.

Todos os resultados obtidos neste trabalho estão disponíveis no Apêndice A, ao final deste artigo.

6 Conclusão

A classifição timbrística de forma automatizada é uma área de interesse, dado o fato de englobar não somente a música e a computação, mas também a matemática e dependendo da aplicação, até mesmo a biologia. Em relação a este trabalho, a classificação automática de timbres de instrumentos pode ser utilizada em plataformas bastante conhecidas como o Youtube e Spotify para criar sistemas de recomendação musical.

Em relação aos resultados obtidos anteriormente, ambos ficaram dentro do esperado. A acurácia do algoritmo implementado ficou bastante semelhante ao fornecido pela Sklearn. Vale ressaltar que as features fornecidas pelo MFCC foram mais eficazes para o algoritmo implementado do que para o código da Sklearn. Desta forma, a adição das características advindas do MFCC aumentaram a eficácia do algoritmo da Sklearn em aproximadamente 3,2% e em 3,4% para o algoritmo implementado.

Em relação aos resultados posteriores, foi observado a necessidade do armazenamento dos dados pré-processados, visto que ao resolver um problema de classificação, é necessário realizar muitos testes para se chegar a uma acurácia satisfatória. Ou seja, a realização das mesmas tarefas de pré-processamento a cada vez em que o teste é executado, é algo que poderia ser evitado através de um sistema de arquivos para armazená-los. Desta forma é necessário apenas carregá-los para que os testes sejam realizados, de maneira mais rápida e eficiente.

Sobre os resultados finais obtidos, a acurácia referente à classificação timbrística foi bastante satisfatória, chegando a aproximadamente 85% de acerto, sendo bem superior se comparado ao Naive Bayes. Além disso, a classificação baseada em um sistema de votos/ensemble provou ser uma técnica bastante eficaz, dado que a combinação de três classificadores geralmente tende a ser mais promissora se comparada ao uso de apenas um classificador.

Após o desenvolvimento deste trabalho, foi observado que nem sempre as *features* coletadas a partir da média e do desvio-padrão, sob resultados de um descritor, serão relevantes. No caso do MFCC, por exemplo, ao mesmo tempo em que a média resultou na *feature* mais relevante para classificação, o desvio-padrão resultou na pior.

Por fim, vale ressaltar que a complexidade do problema da divisão de dois instrumentos distintos é bastante influenciada pelos tipos de instrumentos a serem analisados. A tarefa de distinguir uma bateria de uma flauta, por exemplo, pode ser mais fácil do que separar uma guitarra e um baixo, visto que a bateria é um instrumento de percussão e a flauta um instrumento de sopro, enquanto guitarra e baixo são instrumentos de cordas, com sons semelhantes, o que dificulta a classificação automática.

Referências

[1] Roberto Piassi Passos Bodo and Marcelo Gomes de Queiroz. Três abordagens para similaridade musical utilizando melodia, ritmo e timbre. Master's thesis, Universidade Federal de São Paulo, São Paulo, 2018.

[2] Marcelo Knobel. Física da fala e da audição, 2017.

[3] Harmonia e Improviso. Aspectos físicos do som, Outubro 2019. [Online;].

[4] Martin Vetterli Paolo Prandoni. *Signal Processing for Communications (Communication and Information Sciences)*. EPFL Press, 2008.

[5] Udo Zölzer. *Digital Audio Signal Processing*. John WIley and Sons, 2008.

[6] Jay Hodgson. *Understanding Records: A Field Guide To Recording Practice*. Continuum, 2010.

[7] Ivan Eiji Simurra. A utilização descritores de áudio à análise e composição musical assistidas por computador: um estudo de caso na obra labori ruinae, 2016.

[8] Ivan Eiji Simurra. Análise musical assistida por descritores de Áudio: um estudo de caso da obra reflexões de jônatas manzolli, 2015.

[9] F. Emily. A importância do timbre, 2011.

[10] Aurélio Buarque de Holanda Ferreira. *Novo Aurelio Seculo XXI - O Dicionário Da Língua Portuguesa*. Nova Fronteira, 1999.

[11] Simplifying Theory. Timbre definition, Outubro 2019. [Online; Article].

[12] Paul M Brossier. *Automatic annotation of musical audio for interactive applications*. PhD thesis, University of London, 2006.

[13] Jamie Bullock and UCEB Conservatoire. Libxtract: a lightweight library for audio feature extraction. In *ICMC*, 2007.

[14] George Tzanetakis and Perry Cook. Marsyas: A framework for audio analysis. *Organised sound*, 4(3):169–175, 2000.

[15] Cory McKay, Rebecca Fiebrink, Daniel McEnnis, Beinan Li, and Ichiro Fujinaga. Ace: A framework for optimizing music classification. In *ISMIR*, pages 42–49, 2005.

[16] Xavier Amatriain, Jordi Massaguer, David Garcia, and Ismael Mosquera. The clam annotator: A cross-platform audio descriptors editing tool. In *ISMIR*, pages 426–429, 2005.

[17] Cory McKay and Ichiro Fujinaga. jmir: Tools for automatic music classification. In *ICMC*, 2009.

[18] Chris Cannam, Christian Landone, and Mark Sandler. Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1467–1468. ACM, 2010.

[19] Olivier Lartillot, Petri Toiviainen, and Tuomas Eerola. A matlab toolbox for music information retrieval. In *Data analysis, machine learning and applications*, pages 261–268. Springer, 2008.

[20] SourceForge Editorial. jaudio 1.0 feature appendix. *SourceForge*, 2018.

[21] Music Information Retrieval Conference. Spectral features, Outubro 2015. [Online; Conference].

[22] Sound Analysis Pro Editorial. Wiener entropy, Outubro 2014. [Online; chapter 4].

[23] Luscinia. Spectrogram window, Outubro 2017. [Online; Parameters].

[24] Geoffroy Peeters. A large set of audio features for sound description. *Ircam - Analysis/Synthesis Team*, 2004.

[25] N. Fakotakis e G. Kokkinakis T. Ganchev. Comparative evaluation of various mfcc implementations on the speaker verification task. *10th International Conference on Speech and Computer (SPECOM 2005)*, 2005.

[26] G. Kokkinakis T. Ganchev, N. Fakotakis. Comparative evaluation of various mfcc implementations on the speaker verification. *0th International Conference on Speech and Computer (SPECOM 2005)*, 1:191–194, 2005.

[27] Harry Zhang. The optimality of naive bayes. University Article, 2005.

[28] André Prisco e Eduardo Borge Giancarlo Lucca, Igor Pereira. Uma implementação do algoritmo naive bayes para classificação de texto. University Article, 2005.

[29] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. 2006.

[30] T. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[31] Python Software Foundation. What is python?, 2007.

[32] Stack Overflow. Developer survey results 2018. *Stack Overflow Annual Report*, 2018.

[33] Darryl K. Taft. Jetbrains strikes python developers with pycharm 1.0 ide. *eWeek*, s/n(s/n), Outubro 2010.

[34] Inc JetBrains. Pycharm plugins. *PyCharm Blog*, s/n(s/n), Janeiro 2010.

[35] Fernando Pérez. Jupyter notebook: O que é? *Dados e Decisões*, 2018.

[36] Dawen Liang Daniel Ellis Matt McVicar Eric Battenberg e Oriol Nieto Brian McFee, Colin Raffel. librosa: Audio and music signal analysis in python. *14th Python in Science COnference*, s/n(s/n), 2015.

[37] Librosa Development Team. librosa: tutorial. *Librosa Github*, s/n(s/n), 2013.

[38] Python Software Foundation. Audiolib 1.0.0b1, 2018.

[39] Scikit-Learn Organization. *Release History*, 7 2019. Rev. 3.

[40] Eli Bressert. *SciPy and NumPy*. O'Reilly Media, 2012.

[41] Yotam Mann. The nsynth dataset, Abril 2017. [Online; NSynth Dataset].

[42] Towards data science: The complete guide to decision trees. Accessed: 2019-12-04.

[43] Machine Learning Beyond Deep Learning Árvores de decisão. `https://medium.com/machine-learning-beyond-deep-learning/\%C3\%A1rvores-de-decis\%C3\%A3o-3f52f6420b69`. Accessed: 2019-12-04.

[44] Márcio Porto Basgalupp. *Árvores de Decisão*. PhD thesis, Universidade Estadual de Campinas - UNICAMP, 2010.

[45] Lucid chart: Como criar um diagrama de árvore de decisão. Accessed: 2019-12-04.

[46] Ana Carolina Lorena and André C. P. L. F. de Carvalho. Uma introdução às support vector machines. University Article, 2007.

[47] Python programmin tutorial: Linear svc machine learning svm example with python. Accessed: 2019-12-04.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[49] IntelliPaat which one is better: Linearsvc or svc? Accessed: 2019-12-04.

[50] A. Caponnetto L. Rosasco, E. De Vito and M. Piana. Are loss functions all the same? University Article, 2003.

[51] Towards data science: A guide to ensemble learning: Increase your accuracy by combining model outputs. Accessed: 2019-12-04.

[52] Towards Data Science ensemble learning to improve machine learning results: How ensemble methods work: bagging, boosting and stacking. `https://blog.statsbot.co/ensemble-learning-d1dcd548e936`. Accessed: 2019-12-04.

[53] Peter Norvig. *Inteligência Artificial*. GEN LTC, 2013.

[54] Medium corporation:como criar k-fold cross-validation na mão em python. Accessed: 2019-12-04.

[55] Towards data science: Why and how to cross validate a model? Accessed: 2019-12-04.

[56] Towards Data Science cross validation: A beginner's guide. https://towardsdatascience.com/cross-validation-a-beginners-guide-5b8ca04962cd. Accessed: 2010-09-30.

[57] Medium corporation: Como saber se seu modelo de machine learning está funcionando mesmo. Accessed: 2019-12-04.

[58] Minerando Dados: feature selection a bala de prata. `https://minerandodados.com.br/feature-selection-bala-de-prata/`. Accessed: 2019-12-05.

[59] G. E. P. Box. Non-normality and tests on variances. University Article, 1953.

Apêndice A

```
###################################
# Using cross-validation( k = 5 ) #
###################################

# NB:      Naive Bayes (Sklearn)   #
# SVC:     Suport Vector Class.    #
# DecTree: Decision Tree (Entropy) #
# NBImp:   Naive Bayes Implemented #
# Emsemb:  Emsemble (Bagging)      #

###########################
# Number of Features = 2 #
###########################
k = 1
NB:      67.9 %
SVC:     65.6 %
DecTree: 84.6 %
NBImp:   67.9 %
Emsemb:  78.9 %

k = 2
NB:      70.2 %
SVC:     67.6 %
DecTree: 85.7 %
NBImp:   70.2 %
Emsemb:  79.5 %

k = 3
NB:      65.8 %
SVC:     62.0 %
DecTree: 84.4 %
NBImp:   65.8 %
Emsemb:  76.3 %

k = 4
NB:      68.3 %
SVC:     65.4 %
DecTree: 85.1 %
NBImp:   67.8 %
Emsemb:  78.8 %

k = 5
NB:      65.0 %
SVC:     64.9 %
DecTree: 84.8 %
NBImp:   65.0 %
Emsemb:  77.9 %

###########################
# Number of Features = 3 #
###########################
k = 1
NB:      67.9 %
SVC:     71.3 %
DecTree: 90.8 %
NBImp:   68.4 %
Emsemb:  80.2 %

k = 2
NB:      69.8 %
SVC:     76.0 %
DecTree: 92.0 %
NBImp:   69.6 %
Emsemb:  85.1 %

k = 3
NB:      71.9 %
SVC:     76.2 %
DecTree: 94.2 %
NBImp:   72.1 %
Emsemb:  87.8 %

k = 4
NB:      71.4 %
SVC:     72.8 %
DecTree: 92.2 %
NBImp:   71.4 %
Emsemb:  84.0 %

k = 5
NB:      68.6 %
SVC:     73.7 %
DecTree: 91.1 %
NBImp:   69.4 %
Emsemb:  84.5 %

###########################
# Number of Features = 5 #
###########################
k = 1
NB:      77.1 %
SVC:     81.1 %
DecTree: 92.3 %
NBImp:   76.0 %
Emsemb:  83.8 %

k = 2
NB:      74.8 %
SVC:     81.2 %
DecTree: 91.5 %
NBImp:   74.9 %
Emsemb:  82.5 %

k = 3
NB:      72.1 %
SVC:     78.9 %
DecTree: 91.9 %
NBImp:   72.4 %
Emsemb:  82.3 %

k = 4
NB:      74.7 %
SVC:     79.9 %
DecTree: 93.6 %
NBImp:   73.7 %
Emsemb:  82.2 %
```

```
k = 5
NB:       73.9 %
SVC:      80.3 %
DecTree: 92.8 %
NBImp:    73.8 %
Emsemb:   82.9 %

############################
# Number of Features = 6 #
############################
k = 1
NB:       79.2 %
SVC:      80.5 %
DecTree: 91.7 %
NBImp:    79.4 %
Emsemb:   83.3 %

k = 2
NB:       78.8 %
SVC:      78.2 %
DecTree: 89.0 %
NBImp:    78.5 %
Emsemb:   82.0 %

k = 3
NB:       78.7 %
SVC:      79.8 %
DecTree: 93.3 %
NBImp:    78.5 %
Emsemb:   82.9 %

k = 4
NB:       81.1 %
SVC:      82.3 %
DecTree: 90.1 %
NBImp:    81.4 %
Emsemb:   85.5 %

k = 5
NB:       79.3 %
SVC:      79.6 %
DecTree: 91.7 %
NBImp:    79.3 %
Emsemb:   82.9 %

############################
# Number of Features = 8 #
############################
k = 1
NB:       78.3 %
SVC:      79.1 %
DecTree: 88.3 %
NBImp:    80.1 %
Emsemb:   83.4 %

k = 2
NB:       81.4 %
SVC:      81.5 %
DecTree: 89.3 %
NBImp:    80.4 %
Emsemb:   84.9 %

k = 3
NB:       78.7 %
SVC:      78.7 %
DecTree: 90.0 %
NBImp:    79.3 %
Emsemb:   83.5 %

k = 4
NB:       79.0 %
SVC:      80.8 %
DecTree: 92.4 %
NBImp:    79.4 %
Emsemb:   85.3 %

k = 5
NB:       79.5 %
SVC:      81.1 %
DecTree: 91.2 %
NBImp:    77.4 %
Emsemb:   85.1 %
```